Содержание

Предисловие 19

Структура книги 20
Изменения в третьем издании 24
Будущее C++ 25
Благодарности 25
Благодарности во втором издании 26
Список литературы 27

Часть I

Краткий обзор языка C++ 29

1. Начинаем 31

1.2. Программа на языке C++ 32
1.2.1. Порядок выполнения инструкций 37
1.3. Директивы препроцессора 38
1.4. Немного о комментариях 41
1.5. Первый взгляд на ввод/вывод 43
1.5.1. Файловый ввод/вывод 45

2. Краткий обзор C++ 47

2.1. Встроенный тип данных “массив” 47
2.2. Динамическое выделение памяти и указатели 50
2.3. Объектный подход 53
2.4. Объектно-ориентированный подход 63
2.5. Использование шаблонов 70
2.6. Использование исключений 76
2.7. Использование пространства имен 80
2.8. Стандартный массив — это вектор 83

Часть II

Основы языка 89

3. Типы данных C++ 91

3.1. Литералы 91
3.2. Переменные 94
3.2.1. Что такое переменная 96
3.2.2. Имя переменной 98
3.2.3. Определение объекта 99

3.3. Указатели ... 101
3.4. Строковые типы ... 106
3.4.1. Встроенный строковый тип ... 106
3.4.2. Класс string ... 108
3.5. Квалификатор const ... 113
3.6. Ссылочный тип ... 116
3.7. Тип bool ... 120
3.8. Перечисления ... 121
3.9. Тип “массив” ... 123
3.9.1. Многомерные массивы ... 126
3.9.2. Взаимосвязь массивов и указателей ... 128
3.10. Класс vector ... 130
3.11. Класс complex ... 134
3.12. Директива typedef ... 134
3.13. Квалификатор volatile ... 135
3.14. Класс pair ... 136
3.15. Типы классов ... 137

4. Выражения ... 147

4.1. Что такое выражение? ... 147
4.2. Арифметические операции ... 149
4.3. Операции сравнения и логические операции ... 151
4.4. Операции присваивания ... 154
4.5. Операции инкремента и декремента ... 157
4.6. Операции с комплексными числами ... 159
4.7. Условное выражение ... 162
4.8. Оператор sizeof ... 163
4.9. Операторы new и delete ... 164
4.10. Оператор “запятая” ... 166
4.11. Побитовые операторы ... 166
4.12. Класс bitset ... 169
4.13. Приоритеты ... 173
4.14. Преобразования типов ... 176
4.14.1. Неявное преобразование типов ... 177
4.14.2. Арифметические преобразования типов ... 178
4.14.3. Явное преобразование типов ... 179
4.14.4. Устаревшая форма явного преобразования ... 183
4.15. Пример: реализация класса Stack ... 184

5. Инструкции ... 188

5.1. Простые и составные инструкции ... 188
5.2. Инструкции объявления ... 189

5.3. Инструкция if ... 192
5.4. Инструкция switch ... 199
5.5. Инструкция цикла for ... 206
5.6. Инструкция while ... 209
5.7. Инструкция do while ... 211
5.8. Инструкция break ... 213
5.9. Инструкция continue ... 214
5.10. Инструкция goto ... 215
5.11. Пример связанного списка ... 216
5.11.1. Обобщенный список ... 233

6. Абстрактные контейнерные типы ... 239

6.1. Система текстового поиска ... 240
6.2. Вектор или список? ... 243
6.3. Как растет вектор? ... 245
6.4. Как определить последовательный контейнер? ... 248
6.5. Итераторы ... 252
6.6. Операции с последовательными контейнерами ... 256
6.6.1. Удаление ... 257
6.6.2. Присваивание и обмен ... 258
6.6.3. Обобщенные алгоритмы ... 258
6.7. Читаем текстовый файл ... 259
6.8. Выделяем слова в строке ... 262
6.9. Обрабатываем знаки препинания ... 267
6.10. Приводим слова к стандартной форме ... 270
6.11. Дополнительные операции со строками ... 273
6.12. Строим отображение позиций слов ... 278
6.12.1. Определение объекта map и заполнение его элементами ... 278
6.12.2. Поиск и извлечение элемента отображения ... 282
6.12.3. Перебор элементов отображения map ... 283
6.12.4. Словарь ... 284
6.12.5. Удаление элементов map ... 286
6.13. Построение набора исключаемых слов ... 287
6.13.1. Определение объекта set и заполнение его элементами ... 287
6.13.2. Поиск элемента ... 288
6.13.3. Перебор элементов множества set ... 289
6.14. Окончательная программа ... 290
6.15. Контейнеры multimap и multiset ... 299
6.16. Стек ... 301
6.17. Очередь и очередь с приоритетами ... 303
6.18. Вернемся к классу iStack ... 304

Часть III

Процедурно-ориентированное программирование 307

7. Функции 309

7.1. Введение 309
7.2. Прототип функции 312
7.2.1. Тип возвращаемого функцией значения 312
7.2.2. Список параметров функции 313
7.2.3. Проверка типов формальных параметров 313
7.3. Передача аргументов 315
7.3.1. Параметры-ссылки 317
7.3.2. Параметры-ссылки и параметры-указатели 319
7.3.3. Параметры-массивы 322
7.3.4. Абстрактные контейнерные типы в качестве параметров 325
7.3.5. Значения параметров по умолчанию 326
7.3.6. Многоточие 328
7.4. Возврат значения 330
7.4.1. Параметры и возвращаемые значения против глобальных объектов 334
7.5. Рекурсия 335
7.6. Встроенные функции 336
7.7. Директива линкования: extern "C" 337
7.8. Функция main(): разбор параметров командной строки 340
7.8.1. Класс для обработки параметров командной строки 347
7.9. Указатели на функции 349
7.9.1. Тип указателя на функцию 350
7.9.2. Инициализация и присваивание 351
7.9.3. Вызов 352
7.9.4. Массивы указателей на функции 353
7.9.5. Параметры и тип возврата 354
7.9.6. Указатели на функции, объявленные как extern "C" 356

8. Область видимости и время жизни 359

8.1. Область видимости 359
8.1.1. Локальная область видимости 361
8.2. Глобальные объекты и функции 365
8.2.1. Объявления и определения 365
8.2.2. Сопоставление объявлений в разных файлах 366
8.2.3. Несколько слов о заголовочных файлах 367
8.3. Локальные объекты 370
8.3.1. Автоматические объекты 371
8.3.2. Регистровые автоматические объекты 372
8.3.3. Статические локальные объекты 372

8.4. Динамически размещаемые объекты 374
8.4.1. Динамическое создание и уничтожение единичных объектов 374
8.4.2. Шаблон auto_ptr 377
8.4.3. Динамическое создание и уничтожение массивов 381
8.4.4. Динамическое создание и уничтожение константных объектов 383
8.4.5. Оператор размещения new 383
8.5. Определения пространства имен 386
8.5.1. Определения пространства имен 387
8.5.2. Оператор разрешения области видимости 390
8.5.3. Вложенные пространства имен 391
8.5.4. Определение члена пространства имен 393
8.5.5. ПОО и члены пространства имен 395
8.5.6. Безымянные пространства имен 396
8.6. Использование членов пространства имен 398
8.6.1. Псевдонимы пространства имен 398
8.6.2. Using-объявления 399
8.6.3. Using-директивы 401
8.6.4. Стандартное пространство имен std 404

9. Перегруженные функции 407

9.1. Объявления перегруженных функций 407
9.1.1. Зачем нужно перегружать имя функции 408
9.1.2. Как перегрузить имя функции 408
9.1.3. Когда не надо перегружать имя функции 410
9.1.4. Перегрузка и область видимости 411
9.1.5. Директива extern "C" и перегруженные функции 415
9.1.6. Указатели на перегруженные функции 416
9.1.7. Безопасное к типу линкование 417
9.2. Три шага разрешения перегрузки 418
9.3. Преобразования типов аргументов 421
9.3.1. Подробнее о точном соответствии 422
9.3.2. Подробнее о повышении типов 427
9.3.3. Подробнее о стандартном преобразовании 429
9.3.4. Ссылки 432
9.4. Детали разрешения перегрузки функций 435
9.4.1. Функции-кандидаты 436
9.4.2. Подходящие функции 440
9.4.3. Наиболее подходящая функция 442
9.4.4. Аргументы со значениями по умолчанию 447

10. Шаблоны функций 449

10.1. Определение шаблона функции 449
10.2. Конкретизация шаблона функции 456

10.3. Вывод аргументов шаблона A ... 459
10.4. Явное задание аргументов шаблона A ... 463
10.5. Модели компиляции шаблонов A ... 466
10.5.1. Модель компиляции с включением ... 467
10.5.2. Модель компиляции с разделением ... 468
10.5.3. Явные объявления конкретизации ... 469
10.6. Явная специализация шаблона A ... 471
10.7. Перегрузка шаблонов функций A ... 476
10.8. Разрешение перегрузки при конкретизации A ... 478
10.9. Разрешение имен в определениях шаблонов A ... 485
10.10. Пространства имен и шаблоны функций A ... 491
10.11. Пример шаблона функции ... 495

11. Обработка исключений ... 498

11.1. Возбуждение исключения ... 498
11.2. try-блок ... 501
11.3. Перехват исключений ... 505
11.3.1. Объекты-исключения ... 506
11.3.2. Раскрутка стека ... 509
11.3.3. Повторное возбуждение исключения ... 510
11.3.4. Перехват всех исключений ... 511
11.4. Спецификации исключений ... 513
11.4.1. Спецификации исключений и указатели на функции ... 516
11.5. Исключения и вопросы проектирования ... 517

12. Обобщенные алгоритмы ... 520

12.1. Краткий обзор ... 520
12.2. Использование обобщенных алгоритмов ... 524
12.3. Объекты-функции ... 533
12.3.1. Предопределенные объекты-функции ... 536
12.3.2. Арифметические объекты-функции ... 537
12.3.3. Сравнительные объекты-функции ... 538
12.3.4. Логические объекты-функции ... 538
12.3.5. Адаптеры функций для объектов-функций ... 539
12.3.6. Реализация объекта-функции ... 540
12.4. Еще раз об итераторах ... 541
12.4.1. Итераторы вставки ... 542
12.4.2. Обратные итераторы ... 543
12.4.3. Потоковые итераторы ... 544
12.4.4. Итератор istream_iterator ... 545
12.4.5. Итератор ostream_iterator ... 546
12.4.6. Пять категорий итераторов ... 547

12.5. Обобщенные алгоритмы ... 549
12.5.1. Алгоритмы поиска ... 550
12.5.2. Алгоритмы сортировки и упорядочения ... 550
12.5.3. Алгоритмы удаления и подстановки ... 551
12.5.4. Алгоритмы перестановки ... 551
12.5.5. Численные алгоритмы ... 551
12.5.6. Алгоритмы генерирования и модификации ... 552
12.5.7. Алгоритмы сравнения ... 552
12.5.8. Алгоритмы работы с множествами ... 552
12.5.9. Алгоритмы работы с кучей ... 552
12.6. Когда нельзя использовать обобщенные алгоритмы ... 552
12.6.1. Операция list_merge() ... 553
12.6.2. Операция list::remove() ... 554
12.6.3. Операция list::remove_if() ... 554
12.6.4. Операция list::reverse() ... 554
12.6.5. Операция list::sort() ... 554
12.6.6. Операция list::splice() ... 554
12.6.7. Операция list::unique() ... 555

Часть IV

Объектное программирование 557

13. Классы ... 559

13.1. Определение класса ... 559
13.1.1. Данные-члены ... 560
13.1.2. Функции-члены ... 561
13.1.3. Доступ к членам ... 563
13.1.4. Друзья ... 564
13.1.5. Объявление и определение класса ... 565
13.2. Объекты классов ... 566
13.3. Функции-члены класса ... 569
13.3.1. Когда использовать встроенные функции-члены ... 569
13.3.2. Доступ к членам класса ... 571
13.3.3. Закрытые и открытые функции-члены ... 572
13.3.4. Особые функции-члены ... 574
13.3.5. Функции-члены с квалификаторами const и volatile ... 575
13.3.6. Объявление mutable ... 578
13.4. Неявный указатель this ... 579
13.4.1. Когда использовать указатель this ... 581
13.5. Статические члены класса ... 584
13.5.1. Статические функции-члены ... 588

13.6. Указатель на член класса ... 590
13.6.1. Тип члена класса ... 592
13.6.2. Работа с указателями на члены класса ... 595
13.6.3. Указатели на статические члены класса ... 596
13.7. Объединение — класс, экономящий память ... 598
13.8. Битовое поле — член, экономящий память ... 603
13.9. Область видимости класса ... 604
13.9.1. Разрешение имен в области видимости класса ... 608
13.10. Вложенные классы ... 611
13.10.1. Разрешение имен в области видимости вложенного класса ... 617
13.11. Классы как члены пространства имен ... 621
13.12. Локальные классы ... 624

14. Инициализация, присваивание и уничтожение класса ... 627

14.1. Инициализация класса ... 627
14.2. Конструктор класса ... 629
14.2.1. Конструктор по умолчанию ... 635
14.2.2. Ограничение прав на создание объекта ... 637
14.2.3. Копирующий конструктор ... 637
14.3. Деструктор класса ... 639
14.3.1. Явный вызов деструктора ... 642
14.3.2. Опасность увеличения размера программы ... 643
14.4. Массивы и векторы объектов ... 645
14.4.1. Инициализация массива, размещенного в куче ... 647
14.4.2. Вектор объектов класса ... 649
14.5. Список инициализации членов ... 651
14.6. Почленная инициализация ... 657
14.6.1. Инициализация члена, являющегося объектом класса ... 660
14.7. Почленное присваивание ... 663
14.8. Соображения эффективности ... 665

15. Перегруженные операторы и преобразования, определенные пользователем ... 671

15.1. Перегрузка операторов ... 671
15.1.1. Члены и не члены класса ... 674
15.1.2. Имена перегруженных операторов ... 678
15.1.3. Разработка перегруженных операторов ... 679
15.2. Друзья ... 680
15.3. Оператор = ... 683
15.4. Оператор [] ... 685
15.5. Оператор () ... 686
15.6. Оператор –> ... 687
15.7. Операторы ++ и –– ... 690

15.8. Операторы new и delete 693
15.8.1. Операторы new[ ] и delete[ ] 697
15.8.2. Оператор размещения new() и оператор delete() 699
15.9. Определенные пользователем преобразования 701
15.9.1. Конвертеры 705
15.9.2. Конструктор как конвертер 708
15.10. Выбор преобразования 710
15.10.1. Еще раз о разрешении перегрузки функций 713
15.10.2. Функции-кандидаты 714
15.10.3. Функции-кандидаты для вызова функции в области видимости класса 716
15.10.4. Ранжирование последовательностей преобразований, определенных пользователем 717
15.11. Разрешение перегрузки и функции-члены 721
15.11.1. Объявления перегруженных функций-членов 722
15.11.2. Функции-кандидаты 723
15.11.3. Подходящие функции 723
15.12. Разрешение перегрузки и операторы 726
15.12.1. Операторные функции-кандидаты 727
15.12.2. Подходящие функции 731
15.12.3. Неоднозначность 732

16. Шаблоны классов 735

16.1. Определение шаблона класса 735
16.1.1. Определения шаблонов классов Queue и QueueItem 741
16.2. Конкретизация шаблона класса 743
16.2.1. Аргументы шаблона для параметров-констант 747
16.3. Функции-члены шаблонов классов 751
16.3.1. Функции-члены шаблонов Queue и QueueItem 752
16.4. Объявления друзей в шаблонах классов 754
16.4.1. Объявления друзей в шаблонах Queue и QueueItem 756
16.5. Статические члены шаблонов класса 759
16.6. Вложенные типы шаблонов классов 761
16.7. Шаблоны-члены 764
16.8. Шаблоны классов и модель компиляции 768
16.8.1. Модель компиляции с включением 770
16.8.2. Модель компиляции с разделением 770
16.8.3. Явные объявления конкретизации 773
16.9. Специализации шаблонов классов 774
16.10. Частичные специализации шаблонов классов 778
16.11. Разрешение имен в шаблонах классов 779
16.12. Пространства имен и шаблоны классов 782
16.13. Шаблон класса Array 783

Часть V

Объектно-ориентированное программирование 791

17. Наследование и подтипизация классов 793

17.1. Определение иерархии классов 796
17.1.1. Объектно-ориентированное проектирование 798
17.2. Идентификация членов иерархии 803
17.2.1. Определение базового класса 804
17.2.2. Определение производных классов 808
17.2.3. Резюме 811
17.3. Доступ к членам базового класса 813
17.4. Конструирование базового и производного классов 819
17.4.1. Конструктор базового класса 821
17.4.2. Конструктор производного класса 822
17.4.3. Альтернативная иерархия классов 823
17.4.4. Отложенное обнаружение ошибок 826
17.4.5. Деструкторы 827
17.5. Виртуальные функции в базовом и производном классах 829
17.5.1. Виртуальный ввод/вывод 830
17.5.2. Чисто виртуальные функции 835
17.5.3. Статический вызов виртуальной функции 837
17.5.4. Виртуальные функции и аргументы по умолчанию 839
17.5.5. Виртуальные деструкторы 841
17.5.6. Виртуальная функция eval() 842
17.5.7. Почти виртуальный оператор new 847
17.5.8. Виртуальные функции, конструкторы и деструкторы 849
17.6. Почленная инициализация и присваивание A 851
17.7. Управляющий класс UserQuery 856
17.7.1. Определение класса UserQuery 860
17.8. Соберем все вместе 864

18. Множественное и виртуальное наследование 871

18.1. Готовим сцену 871
18.2. Множественное наследование 875
18.3. Открытое, закрытое и защищенное наследование 882
18.3.1. Наследование и композиция 884
18.3.2. Открытие отдельных членов 885
18.3.3. Защищенное наследование 886
18.3.4. Композиция объектов 887
18.4. Область видимости класса и наследование 889
18.4.1. Область видимости класса при множественном наследовании 892

18.5. Виртуальное наследование 897
18.5.1. Объявление виртуального базового класса 899
18.5.2. Специальная семантика инициализации 901
18.5.3. Порядок вызова конструкторов и деструкторов 903
18.5.4. Видимость членов виртуального базового класса 905
18.6. Пример множественного виртуального наследования 908
18.6.1. Порождение класса, контролирующего выход за границы массива 911
18.6.2. Порождение класса отсортированного массива 913
18.6.3. Класс массива с множественным наследованием 918

19. Применение наследования в C++ 921

19.1. Идентификация типов во время выполнения 921
19.1.1. Оператор dynamic_cast 922
19.1.2. Оператор typeid 927
19.1.3. Класс type_info 929
19.2. Исключения и наследование 932
19.2.1. Исключения, определенные как иерархии классов 932
19.2.2. Возбуждение исключения типа класса 933
19.2.3. Обработка исключения типа класса 934
19.2.4. Объекты-исключения и виртуальные функции 937
19.2.5. Раскрутка стека и вызов деструкторов 938
19.2.6. Спецификации исключений 940
19.2.7. Конструкторы и функциональные try-блоки 942
19.2.8. Иерархия классов исключений в стандартной библиотеке C++ 944
19.3. Разрешение перегрузки и наследование 948
19.3.1. Функции-кандидаты 948
19.3.2. Подходящие функции и последовательности пользовательских преобразований 952
19.3.3. Наиболее подходящая функция 953

20. Библиотека iostream 957

20.1. Оператор вывода << 961
20.2. Ввод 965
20.2.1. Строковый ввод 969
20.3. Дополнительные операторы ввода/вывода 975
20.4. Перегрузка оператора вывода 981
20.5. Перегрузка оператора ввода 985
20.6. Файловый ввод/вывод 987
20.7. Состояния потока 996
20.8. Строковые потоки 998
20.9. Состояние формата 1000
20.10. Строго типизированная библиотека 1008

Приложение

Обобщенные алгоритмы в алфавитном порядке 1009

accumulate() 1011
adjacent_difference() 1012
adjacent_find() 1013
binary_search() 1014
copy() 1014
copy_backward() 1015
count() 1017
count_if() 1018
equal() 1019
equal_range() 1021
fill() 1023
fill_n() 1024
find() 1025
find_if() 1026
find_end() 1027
find_first_of() 1028
for_each() 1030
generate() 1030
generate_n() 1031
includes() 1032
inner_product() 1033
inplace_merge() 1034
iter_swap() 1036
lexicographical_compare() 1036
lower_bound() 1038
max() 1039
max_element() 1039
min() 1040
min_element() 1040
merge() 1041
mismatch() 1042
next_permutation() 1044
nth_element() 1045
partial_sort() 1046
partial_sort_copy() 1047
partial_sum() 1048
partition() 1049
prev_permutation() 1051
random_shuffle() 1051
remove() 1052

remove_copy() ... 1053
remove_if() ... 1054
remove_copy_if() ... 1054
replace() ... 1055
replace_copy() ... 1055
replace_if() ... 1056
replace_copy_if() ... 1056
reverse() ... 1058
reverse_copy() ... 1058
rotate() ... 1059
rotate_copy() ... 1059
search() ... 1060
search_n() ... 1061
set_difference() ... 1062
set_intersection() ... 1063
set_symmetric_difference() ... 1063
set_union() ... 1064
sort() ... 1066
stable_partition() ... 1066
stable_sort() ... 1067
swap() ... 1068
swap_ranges() ... 1069
transform() ... 1070
unique() ... 1071
unique_copy() ... 1072
upper_bound() ... 1073
Алгоритмы для работы с кучей ... 1074
make_heap() ... 1074
pop_heap() ... 1075
push_heap() ... 1075
sort_heap() ... 1075

Посвящается Бет,
благодаря которой стала возможна эта книга,
да и все остальное тоже

Посвящается Даниэлю и Анне,
которым открыты все возможности

Стенли Липпман

Посвящается Марку и маме,
за их любовь и поддержку безо всяких условий

Жози Лажойе

Предисловие

Между выходом второго и третьего издания “C++, вводный курс” произошло довольно много событий. Одним из самых значительных стало появление международного стандарта. Он не только добавил в язык C++ новые возможности, среди которых обработка исключений, идентификация типов во время выполнения, пространство имен, встроенный булевский тип данных, новый синтаксис приведения типов, но также существенно изменил и расширил уже имевшиеся — шаблоны, механизм классов, поддерживающий объектную и объектно-ориентированную парадигму программирования, вложенные типы и разрешение перегруженных функций. Еще более важным событием стало включение в состав стандарта C++ обширной библиотеки, содержащей, в частности, то, что ранее называлось Standard Template Library (STL). В эту стандартную библиотеку входят новый тип `string`, последовательные и ассоциативные контейнеры, такие как `vector`, `list`, `map`, `set`, и обширный набор обобщенных алгоритмов, которые могут применяться ко всем этим типам данных. Появилось не просто много нового материала, нуждающегося в описании, но фактически изменился сам способ мышления при программировании на C++. Короче говоря, можно считать, что C++ изобретен заново, поэтому третье издание нашей книги “C++ для начинающих” полностью переработано.

В третьем издании не только коренным образом поменялся наш подход к C++, изменился и состав авторов. Прежде всего, авторский коллектив удвоился и стал интернациональным, хотя корни его по-прежнему — на североамериканском континенте: Стенли — американец, а Жози — канадка. Двойное авторство отражает деление сообщества программистов C++ на две части: Стенли в настоящее время занимается разработкой прикладных программ на C++ в области трехмерной графики и анимации для Walt Disney Feature Animation, а Жози принимает участие в развитии самого языка C++, являясь председателем рабочей группы по ядру языка в комитете по стандартизации и одним из разработчиков компилятора C++ в IBM Canada Laboratory.

Стенли работает над C++ с 1984 года. Он был одним из членов первоначальной команды, трудившейся в Bell Laboratories под руководством Бьерна Страуструпа — изобретателя языка. Стенли принимал участие в разработке cfront, оригинальной реализации C++, с версии 1.1 в 1986 году до версии 3.0, и возглавлял проект при работе над версиями 2.1 и 3.0. После этого он работал под началом Страуструпа над проектом, посвященным исследованиям объектной модели программной среды разработки на C++.

Жози — член команды, работающей над компилятором C++ в IBM Canada Laboratory на протяжении восьми лет. С 1990 года она входит в состав комитета по стандартизации. Три года она была вице-президентом комитета и четыре года — председателем рабочей группы по ядру языка.

Третье издание “C++, вводный курс” существенно переработано, что отражает не только развитие и расширение языка, но и изменения во взглядах и опыте авторов книги.

Структура книги

“C++, вводный курс” содержит обстоятельное введение в международный стандарт C++. Мы включили в название книги слова “вводный курс”, потому что последовательно придерживались учебного подхода к описанию языка C++, однако название не предполагает упрощенного или облегченного изложения материала. Такие аспекты программирования, как обработка исключений, контейнерные типы, объектно-ориентированный подход и пр., представлены в книге в контексте решения конкретных задач. Правила языка, например разрешение перегруженных функций или преобразования типов в объектно-ориентированном программировании, рассматриваются столь подробно, что во вводном курсе это может показаться неуместным. Но мы уверены, что такое освещение необходимо для практического применения языка. Материал книги не нужно стараться усвоить “за один проход”: мы предполагаем, что читатель будет периодически возвращаться к ранее прочитанным разделам. Если некоторые из них покажутся вам слишком трудными или просто скучными, отложите их на время. (Подозрительные разделы мы помечали значком  .)

Читатель может не знать языка C, хотя некоторое знакомство с каким-либо современным структурным языком программирования было бы полезно. Мы писали книгу, чтобы она стала первым учебником по C++, а не первым учебником по программированию! Чтобы не делать предположений о начальном уровне подготовки, мы начинаем с определения базовых терминов. В первых главах описываются базовые концепции, такие как переменные и циклы, и для некоторых читателей изложение может показаться слишком примитивным, но вскоре оно становится более углубленным.

Основное достоинство C++ заключается в том, что он поддерживает новые способы решения программистских задач. Поэтому, чтобы научиться эффективно использовать C++, недостаточно просто выучить новые синтаксис и семантику. Для более глубокого усвоения в книге рассматриваются разнообразные сквозные примеры. Эти примеры используются как для того, чтобы представить разные средства языка, так и для того, чтобы объяснить, зачем эти средства нужны. Изучая возможности языка в контексте реального примера, мы понимаем, чем полезно то или иное средство, как и где его можно применить при решении задач из реальной жизни. Кроме того, на примерах проще продемонстрировать понятия языка, которые еще детально не рассматривались и излагаются лишь в последующих главах. В начальных главах примеры содержат простые варианты использования базовых понятий C++. Их цель — показать, как можно программировать на C++, не углубляясь в детали проектирования и реализации.

Главы 1 и 2 представляют собой полное введение в язык C++ и его обзор. Назначение первой части — как можно быстрее познакомить читателя с понятиями и средствами данного языка, а также основными принципами написания программ.

По окончании этой части у вас должно сложиться некоторое общее представление о возможностях C++, но вместе с тем вполне может остаться ощущение, что вы *совсем ничего толком не понимаете*. Все нормально: упорядочению ваших знаний как раз и посвящены остальные части книги.

В главе 1 представлены базовые элементы языка: встроенные типы данных, переменные, выражения, инструкции и функции. Мы увидим минимальную законченную программу на C++, обсудим вопросы компиляции, коснемся препроцессора и поддержки ввода/вывода. В этой главе читатель найдет ряд простых, но законченных программ на C++, которые можно откомпилировать и выполнить. Глава 2 посвящена механизму классов и тому, как с его помощью поддержаны парадигмы объектного и объектно-ориентированного программирования. Оба эти подхода иллюстрируются развитием реализации массива как абстрактного типа. Кроме того, приводится краткая информация о шаблонах, пространствах имен, обработке исключений и о поддержке стандартной библиотекой общих контейнерных типов и методов обобщенного (generic) программирования. Материал в этой главе излагается весьма стремительно, и потому некоторым читателям она может показаться трудной. Мы рекомендуем таким читателям просмотреть вторую главу "по диагонали" и вернуться к ней впоследствии.

Фундаментальной особенностью C++ является возможность расширять язык, определяя новые типы данных, которые могут использоваться с тем же удобством и гибкостью, что и встроенные. Первым шагом к овладению этим искусством является знание базового языка. Часть II (главы 3–6) посвящена рассмотрению языка на этом уровне.

В главе 3 представлены встроенные и составные типы, предопределенные в языке, а также типы `string`, `complex` и `vector` из стандартной библиотеки C++. Эти типы составляют основные "кирпичики", из которых строятся все программы. В главе 4 детально освещаются выражения языка — арифметические, условные и присваивания. Инструкции языка, которые являются мельчайшими независимыми единицами C++ программы на C++, приведены в главе 5. Контейнерные типы данных обсуждаются в главе 6. Вместо простого перечисления совокупности поддерживаемых ими операций мы иллюстрируем операции на примере построения системы текстового поиска.

Главы 7–12 (часть III) посвящены *процедурно-ориентированному* программированию на C++. В главе 7 представлен механизм функций. Функция инкапсулирует набор операций, составляющих единую задачу, как, например, `print()`. (Круглые скобки после имени говорят о том, что мы имеем дело с функцией.) Такие понятия, как область видимости и время жизни переменных, рассматриваются в главе 8. Обзор механизма функций продолжен в главе 9: обсуждается перегрузка функций, которая позволяет присвоить одно и то же имя нескольким функциям, выполняющим похожие, но по-разному реализованные операции. Например, можно определить целый набор функций `print()` для печати данных разных типов. В главе 10 представлено понятие шаблона функции и приведены примеры его использования. Шаблон функции предназначен для автоматического генерирования потенциально бесконечного множества экземпляров функций, отличающихся только типами данных.

C++ поддерживает обработку исключений. Об исключении говорят, когда в программе возникает нестандартная ситуация, такая, например, как нехватка свободной памяти. В том месте программы, где это происходит, *возбуждается* исключение, то есть о проблеме ставится в известность вызывающая программа. Какая-то другая функция в программе должна *обработать* исключение, то есть как-то отреагировать на него. Материал об исключениях разбит на две части. В главе 11 описан основной

синтаксис и приведен простой пример, иллюстрирующий возбуждение и обработку исключений типа класса. Поскольку реальные исключения в программах обычно являются объектами некоторой иерархии классов, то мы вернемся к этому вопросу в главе 19, после того как узнаем, что такое объектно-ориентированное программирование.

В главе 12 представлены обширная коллекция обобщенных алгоритмов стандартной библиотеки и способы их применения к контейнерным типам из главы 6, а также к массивам встроенных типов. Эта глава начинается разбором примера, иллюстрирующего, как строится программа с использованием обобщенных алгоритмов. Итераторы, введенные в главе 6, обсуждаются более детально в главе 12, поскольку именно они являются связующим звеном между обобщенными алгоритмами и контейнерными типами. Также мы вводим и иллюстрируем на примерах понятие объекта-функции. Объекты-функции позволяют задавать альтернативную семантику операций, используемых в обобщенных алгоритмах,— например, операций сравнения на равенство или по величине. Детальное описание самих алгоритмов и примеры их использования приводятся в приложении.

Главы 13–16 (часть IV) посвящены *объектному* программированию, то есть использованию механизма классов для создания абстрактных типов данных. С помощью типов данных, описывающих конкретную предметную область, язык C++ позволяет программистам сосредоточиться на решении основной задачи и тратить меньше усилий на второстепенные. Фундаментальные для приложения типы данных могут быть реализованы один раз и использованы многократно, что дает возможность программисту не думать о деталях реализации главной задачи. Инкапсуляция данных значительно упрощает последующее сопровождение и модификацию программы.

В главе 13 основное внимание уделено общим вопросам механизма классов: как определять класс, что такое *сокрытие информации* (разделение открытого интерфейса и скрытой реализации), как определять экземпляры класса и манипулировать ими. Мы также коснемся областей видимости класса, вложенных классов и классов как членов пространства имен.

В главе 14 детально исследуются средства, имеющиеся в C++ для инициализации и уничтожения объектов класса и для присваивания им значений. Для этих целей служат специальные функции-члены, называемые *конструкторами*, *деструкторами* и *копирующими операторами присваивания*. Мы рассмотрим вопрос о почленной инициализации и копировании, а также специальную оптимизацию для этого случая, которая получила название *именованное возвращаемое значение*.

В главе 15 рассмотрена перегрузка операторов применительно к классам. Сначала обсуждаются общие понятия и вопросы проектирования, а затем конкретные операторы, такие как присваивание, доступ по индексу, вызов функции, а также операторы `new` и `delete`, специфичные для классов.

Будет представлено понятие *друзей класса*, имеющих особые права доступа, и объяснено, зачем нужны *друзья*. Будут рассмотрены и определенные пользователями преобразования типов, стоящие за ними концепции и примеры использования. Кроме того, приводятся правила разрешения функций при перегрузке, иллюстрируемые примерами программного кода.

Шаблоны классов — тема главы 16. Шаблон класса можно рассматривать как алгоритм создания экземпляра класса, в котором параметры шаблона подлежат замене

на конкретные значения типов или констант. Например, в шаблоне класса `vector` параметризован тип его элементов. В классе для представления некоторого буфера можно параметризовать не только тип размещаемых элементов, но и размер самого буфера. При разработке сложных механизмов, например в области распределенной обработки данных, могут быть параметризованы практически все интерфейсы: межпроцессной коммуникации, адресации, синхронизации. В главе 16 мы расскажем, как определить шаблон класса, как создать экземпляр класса, подставляя в шаблон конкретные значения, как определить члены шаблона класса (функции-члены, статические члены и вложенные типы) и как следует организовать программу, в которой используются шаблоны классов. Заканчивается эта глава содержательным примером шаблона класса.

Объектно-ориентированному программированию (ООП) и его поддержке в C++ посвящены главы 17–20 (часть V). В главе 17 описываются средства поддержки базовых концепций ООП — наследования и динамического связывания. В ООП между классами, имеющими общие черты поведения, устанавливаются отношения родитель/потомок (или тип/подтип). Вместо того чтобы повторно реализовывать общие характеристики, класс-потомок может унаследовать их от класса-родителя. В класс-потомок (подтип) следует добавить только те детали, которые отличают его от родителя. Например, мы можем определить родительский класс `Employee` (работник) и двух его потомков: `TemporaryEmpl` (временный работник) и `Manager` (начальник), которые наследуют все поведение `Employee`. В них самих реализованы только специфичные для подтипа особенности. Второй аспект ООП, *полиморфизм*, позволяет родительскому классу представлять любого из своих наследников. Скажем, класс `Employee` может адресовать не только объект своего типа, но и объект типа `TemporaryEmpl` или `Manager`. Пóзднее связывание — это способность разрешения операций во время выполнения, то есть выбора нужной операции в зависимости от реального типа объекта. В C++ это реализуется с помощью механизма виртуальных функций.

Итак, в главе 17 представлены базовые черты ООП. В ней мы продолжим начатую в главе 6 работу над системой текстового поиска — спроектируем и реализуем иерархию классов запросов `Query`.

В главе 18 разбираются более сложные случаи наследования — множественное и виртуальное. Шаблон класса из главы 16 получает дальнейшее развитие и становится трехуровневой иерархией с множественным и виртуальным наследованием.

В главе 19 приведено понятие идентификации типа во время выполнения (RTTI — run time type identification). RTTI позволяет программе запросить у полиморфного объекта класса информацию о его типе во время выполнения. Например, мы можем спросить у объекта `Employee`, действительно ли он представляет собой объект типа `Manager`. Кроме того, в главе 19 мы вернемся к исключениям и рассмотрим иерархию классов исключений стандартной библиотеки, приводя примеры построения и использования своей собственной иерархии классов исключений. В этой главе рассматривается также вопрос о разрешении перегруженных функций в случае наследования классов.

В главе 20 подробно обсуждается использование библиотеки ввода/вывода `iostream`. Здесь мы на примерах покажем основные возможности ввода и вывода, расскажем, как определить свои операторы ввода и вывода для класса, как проверять

состояние потока и изменять его, как форматировать данные. Библиотека ввода/вывода представляет собой иерархию классов с множественным и виртуальным наследованием.

Завершается книга приложением, где все обобщенные алгоритмы приведены в алфавитном порядке, с примерами их использования.

При написании книги нередко приходится оставлять в стороне множество вопросов, которые представляются не менее важными, чем вошедшие в книгу. Отдельные аспекты языка — детальное описание того, как работают конструкторы, в каких случаях создаются временные объекты, общие вопросы эффективности — не вписывались во вводный курс. Однако эти аспекты имеют огромное значение при проектировании реальных прикладных программ. Перед тем как взяться за "C++, вводный курс", Стенли написал книгу "Inside the C++ Object Model" [LIPPMAN96a], в которой освещаются именно эти вопросы. В тех местах "C++, вводный курс", где читателю может потребоваться более детальная информация, даются ссылки на разделы указанной книги.

Некоторые части стандартной библиотеки C++ были сознательно исключены из рассмотрения, в частности поддержка национальных языков и численные методы. Стандартная библиотека C++ очень обширна, и все ее аспекты невозможно осветить в одном учебнике. Материал по отсутствующим вопросам вы можете найти в книгах, приведенных в списке литературы ([MUSSER96] и [STRUOSTRUP97]). Наверняка вскоре выйдет еще немало книг, посвященных различным аспектам стандартной библиотеки C++.

Изменения в третьем издании

Все изменения можно разделить на четыре основные категории.

1. Материал, посвященный нововведениям языка: обработке исключений, идентификации типа во время выполнения, пространству имен, встроенному типу bool, новому синтаксису приведения типов.
2. Материал, посвященный стандартной библиотеке C++, в том числе типам `complex`, `string`, `auto_ptr`, `pair`, последовательным и ассоциативным контейнерам (в основном это `list`, `vector`, `map` и `set`) и обобщенным алгоритмам.
3. Коррективы в старом тексте, отражающие улучшения, расширения и изменения, которые новый стандарт C++ привнес в существовавшие ранее средства языка. Примером улучшения может служить использование предваряющих объявлений для вложенных типов, отсутствовавшее ранее. В качестве примера изменения можно привести возможность для экземпляра виртуальной функции производного класса возвращать тип, производный от типа значения, возвращаемого экземпляром той же функции из базового класса. Это изменение поддерживает операцию с классами, которую иногда называют *клонированием*, или *фабрикацией классов* (виртуальная функция `clone()` иллюстрируется в разделе 17.5.7). Пример расширения языка — возможность явно специализировать один или более параметров-типов для шаблонов функций (на самом деле весь механизм шаблонов был радикально расширен — настолько, что его можно назвать новым средством языка).
4. Изменения в подходе к использованию большинства продвинутых средств языка — шаблонов и классов. Стенли считает, что его переход из сравнительно узкого